A GPU-Inspired Soft Processor for High-Throughput Acceleration

Jeffrey Kingyens and J. Gregory Steffan
Electrical and Computer Engineering
University of Toronto

FGPA-Based Acceleration

- **In-socket acceleration platforms**
  - FPGA and CPU on same motherboard
  - Xtremedata, Nallatech, SGI RASC
- **How to program them?**
  - HDL is for experts
  - Behavioural synthesis is limited

XD1000

Can we provide a more familiar programming model?

Potential Solution: Soft Processor

- **Advantages of soft processors:**
  - Familiar, portable, customizable
- **Our Goal: Develop a new S.P. architecture that:**
  - Excels at high-throughput workloads
  - Is naturally capable of high utilization of datapath
- **Challenges:**
  - Memory latency
  - Pipeline latency and hazards
  - Exploiting parallelism
  - Scaling

Inspiration: GPU Architecture

- **Multithreading**
  - Tolerate memory and pipeline latencies
- **Vector instructions**
  - Data-level parallelism, scaling
- **Multiple processors**
  - Scaling

Long-term goal: FPGA-specific design using above

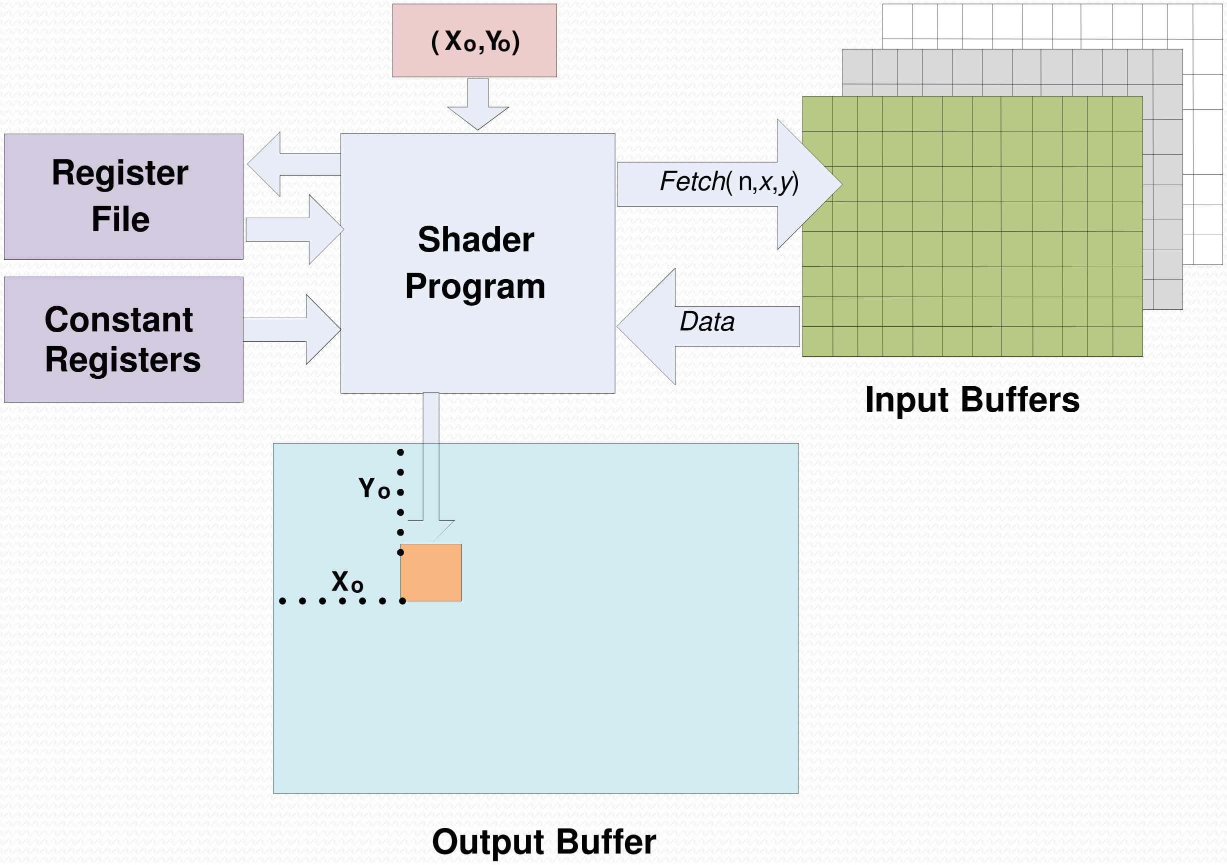
This work: FPGA implementation of a GPU

Overview

- **A GPU-based system**
  - NVIDIA's Cg
  - AMD's CTM r5xx ISA
- **A GPU-inspired architecture**
  - Overcoming port limitations
  - Avoiding stalls
- **Preliminary results**
  - Simulation based on Xtremedata XD1000

A GPU-Based System

GPU Shader Processors



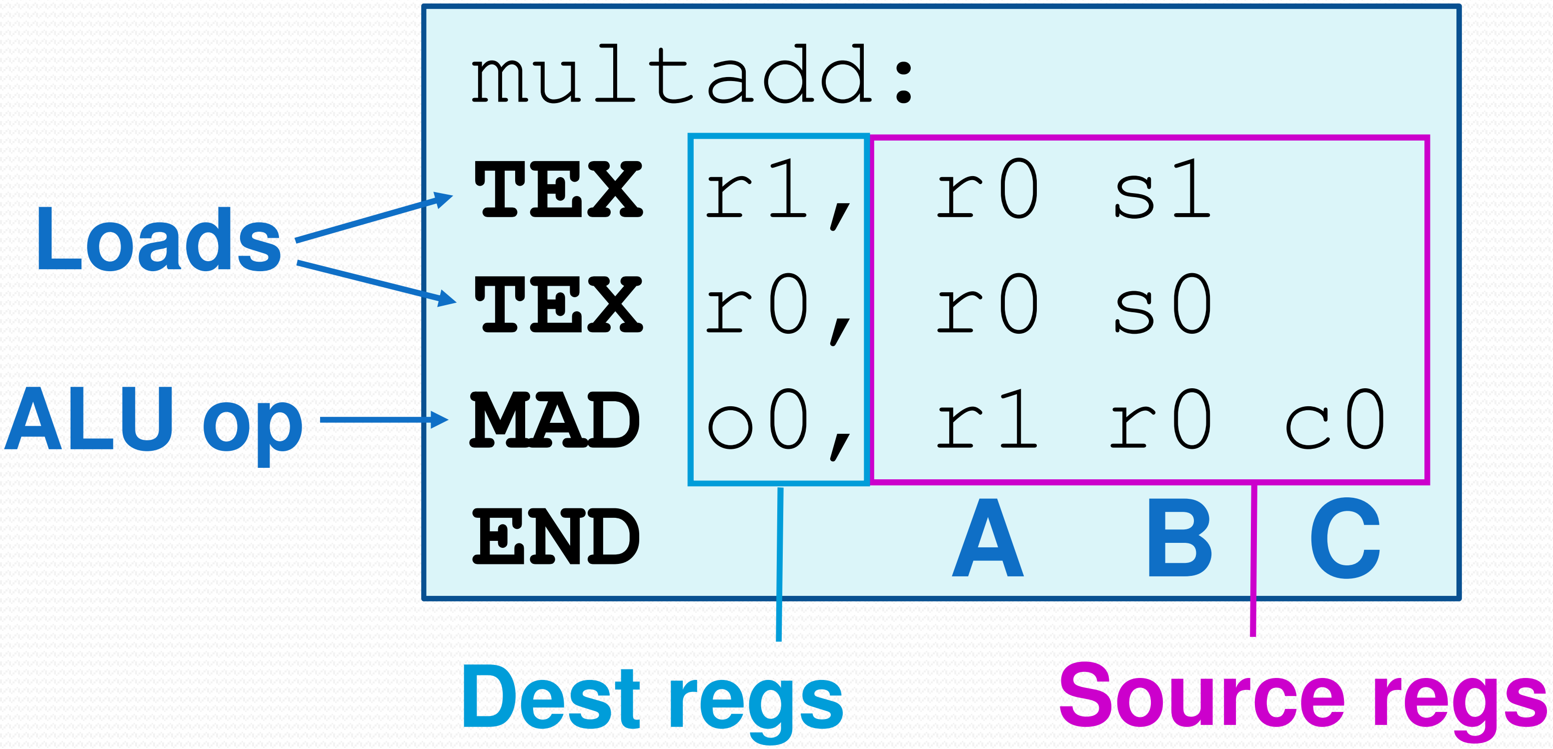
Separate in/out buffers simplify memory coherence

NVIDIA’s Cg Language (C-like)

```
struct data_out {
     float4 sum : COLOR;
};
data_out     multadd(float2 coord : TEXCOORD0,
             uniform sampler2D A: TEXUNIT0,
             uniform sampler2D B: TEXUNIT1)  {
      data_out r;
      float4 offset = {1.0f, 1.0f, 1.0f, 1.0f};
      r.sum = tex2D(A,coord)*tex2D(B,coord)+offset;
      return r;
}
```

Matrix-matrix element-wise multiplication + offset

AMD’s CTM r5xx ISA (simplified)



Loads

ALU op

```
multadd:
TEX r1,  r0 s1
TEX r0,  r0 s0
MAD o0,  r1 r0 c0
END
```

A B C

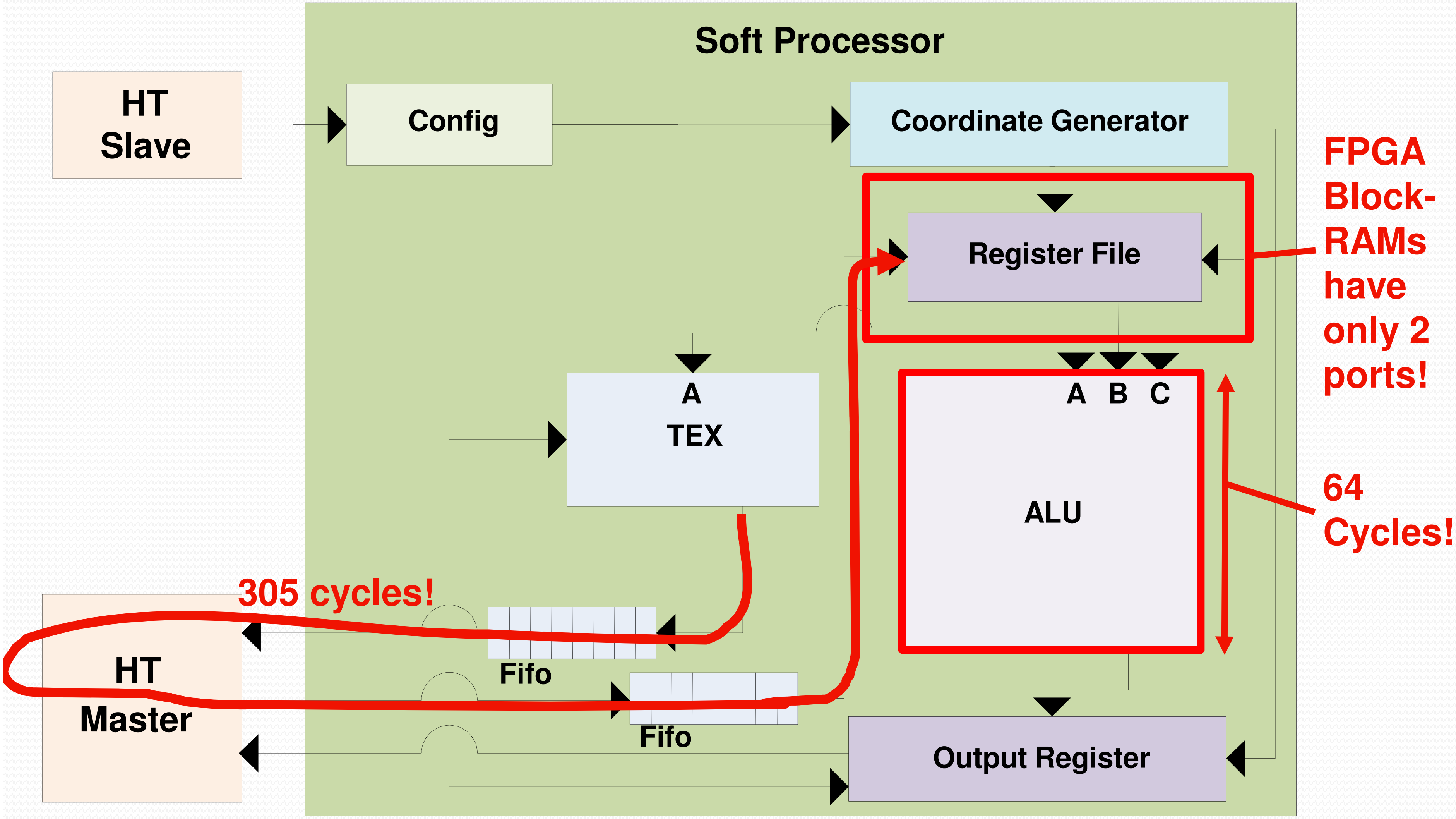
Dest regs

Source regs

Each register is a 4-element vector

A GPU-Inspired Architecture

Soft Processor Architecture



Must tolerate port limitations and latencies

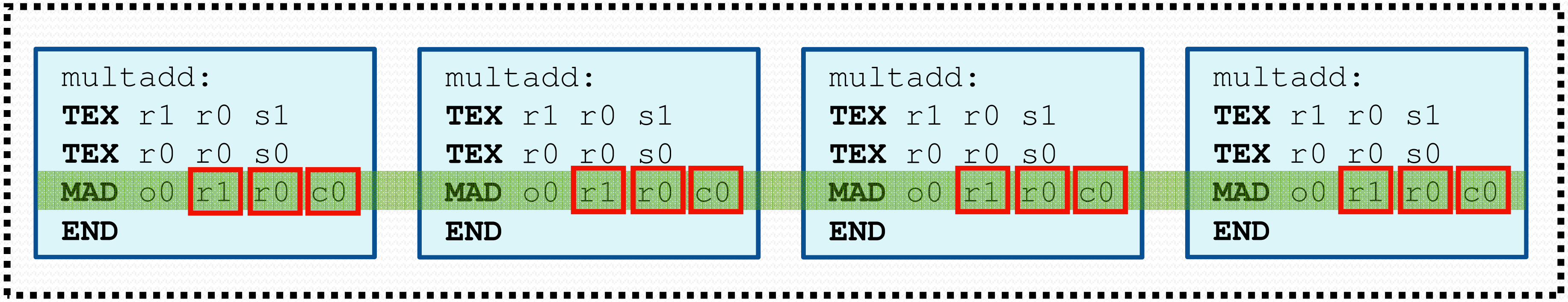
Overcoming Port Limitations

- **Problem: central register file:**
  - Needs four reads and two writes per cycle
  - FPGA block RAMs have only two ports
- **Solution: exploit *symmetry* of threads**
  - Symmetry: every thread executes same inst sequence
  - Group threads into batches of four
  - Fetch operands across batches in lock-step

Only read one operand per thread per cycle

Reading Operands Across Batch

Batch (of 4 threads)



```
multadd:
TEX r1 r0 s1
TEX r0 r0 s0
MAD o0 r1 r0 c0
END
```

```
multadd:
TEX r1 r0 s1
TEX r0 r0 s0
MAD o0 r1 r0 c0
END
```

```
multadd:
TEX r1 r0 s1
TEX r0 r0 s0
MAD o0 r1 r0 c0
END
```

```
multadd:
TEX r1 r0 s1
TEX r0 r0 s0
MAD o0 r1 r0 c0
END
```

Three cycles to read operands:

1) Read A’s

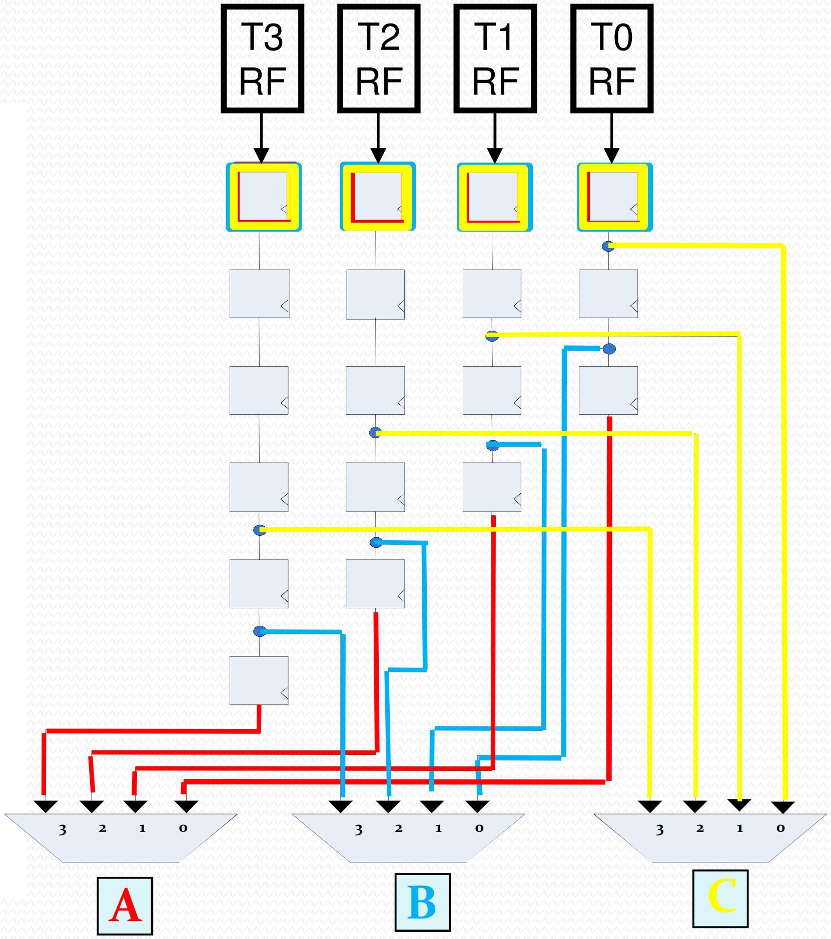
2) Read B’s

3) Read C’s

Only read one operand per thread per cycle

Transposed RegFile Access

Clock Cycle	Inst Phase	Register File Read	ALU Ready



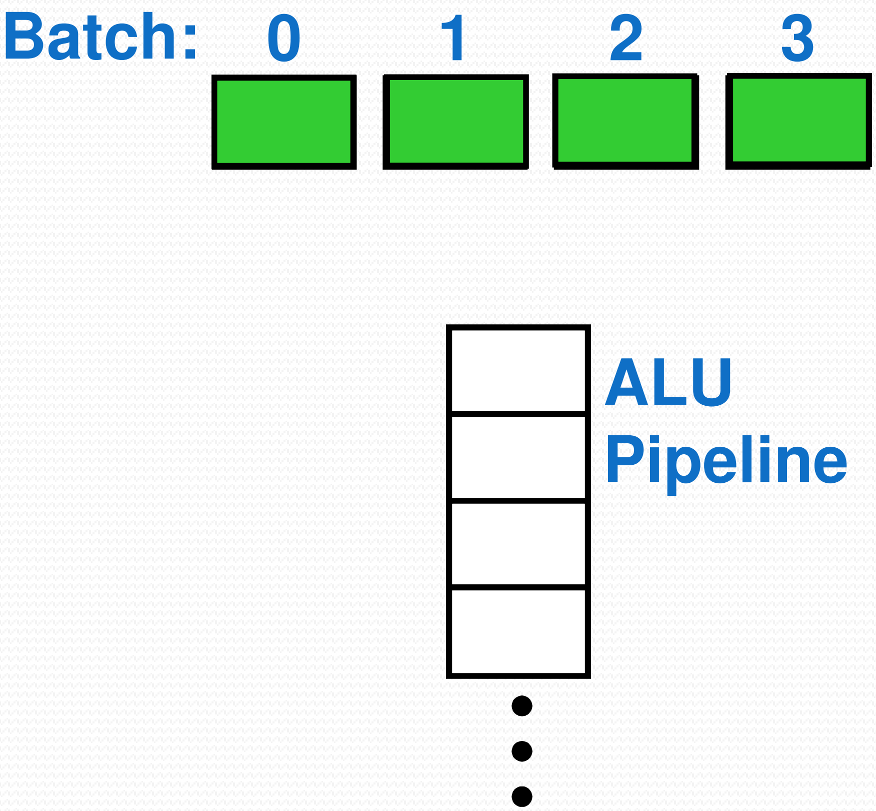
Avoiding ALU Pipeline Bubbles

- **Problem: long pipeline and memory latency**
  - Frequent stalls lead to underutilized ALU datapath
- **Solution: exploit abundance of threads**
  - Store contexts for multiple batches of threads
  - Issue instructions from different batches to hide latencies

Requires logic to issue-from and manage batches

How many batches do we need to avoid bubbles?

Issuing from Multiple Batches



Batch: 0 1 2 3

ALU Pipeline

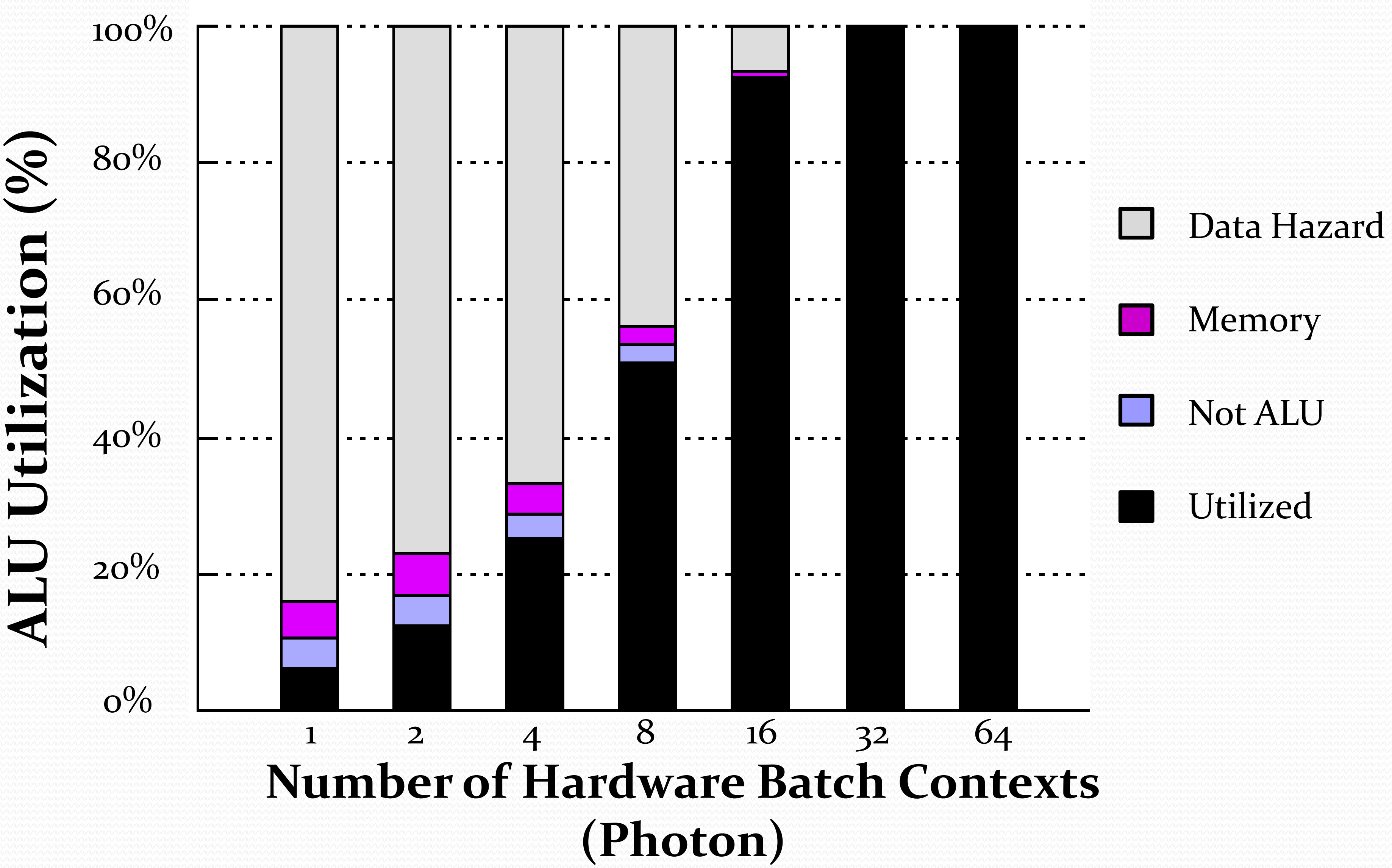
Ideally ALU is fully utilized

Methodology and Results

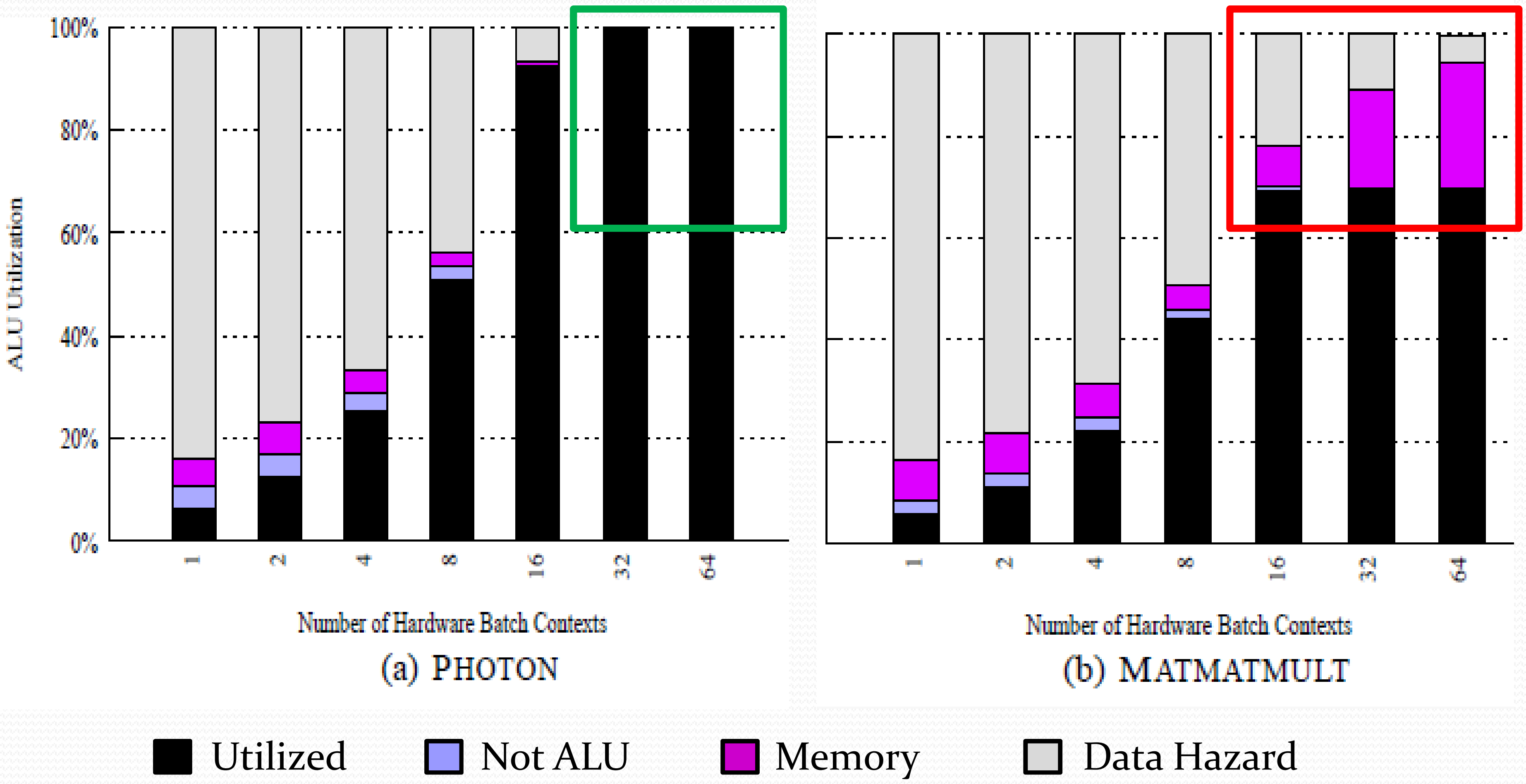
Simulation Methodology

- **SystemC-based simulation**
  - Parameterized to model XD1000
  - Assume conservative 100Mhz soft processor clock
  - Cycle accurate at the block interfaces
  - Models HyperTransport (bandwidth and latency)
    - currently 8-bit HT, capable of 16-bit HT
- **Benchmarks**
  - photon: monte-carlo heat-transfer sim (ALU-intensive)
  - matmatmult: dense matrix multiplication (mem-intensive)

ALU Utilization (8-bit HT)



ALU Utilization (8-bit HT)

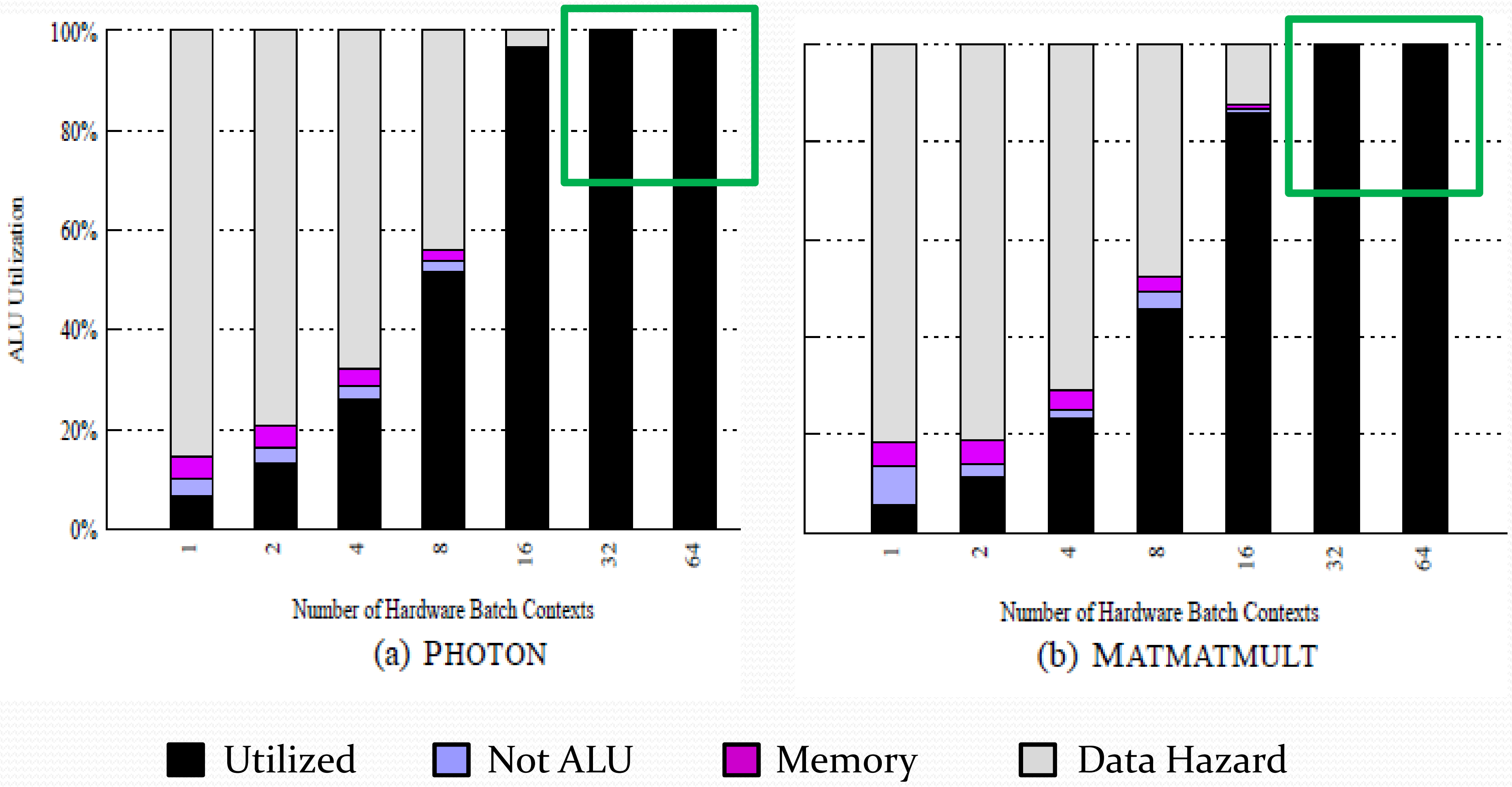


(a) PHOTON

(b) MATMATMULT

Matmatmult is bottlenecked on memory bandwidth

ALU Utilization (16-bit HT)



(a) PHOTON

(b) MATMATMULT

32 batches is sufficient

Conclusions

- **GPU-inspired soft processor architecture**
  - exploits multithreading, vector operations
- **Thread symmetry and batching allows:**
  - tolerating limited block RAM ports
  - tolerating long memory and pipeline latencies
- **32 batches sufficient**
  - to achieve 100% ALU utilization
- **Future work:**
  - customize programming model and arch. to FPGAs
  - exploit longer vectors, multiple CPUs, custom ops